A NOVEL FRAMEWORK FOR MONITORING PARKINSON'S DISEASE PROGRESSION THROUGH VIDEO ANALYSIS AND MACHINE LEARNING

Caroline Zhou[1], Ivan Revilla[2]

[1]The Harker School, 500 Saratoga Ave, San Jose, CA 95129
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Parkinson's disease (PD) is a progressive neurological disorder that necessitates continuous and accurate monitoring for effective management [4]. We propose an innovative system that leverages video analysis and machine learning to predict clinical scores for PD patients. Our system includes a mobile application for recording and uploading videos, a cloud-based server for processing the data, and a machine learning model for analyzing the videos [5]. Key technologies employed include Flutter for the mobile app, Firebase for data storage and authentication, and advanced machine learning models such as Bayesian Ridge and Random Forest regression [6]. Challenges such as variability in video quality and limited dataset diversity were addressed through robust preprocessing techniques and plans to expand the dataset to include more diverse participants. Our experiments demonstrated that Bayesian Ridge and Random Forest regression models achieved high prediction accuracy for clinical scores. The results highlight the system's potential for providing a reliable and user-friendly method for monitoring PD. This comprehensive approach promises significant improvements in patient care and disease management, making it a valuable tool for both patients and healthcare providers.

KEYWORDS

Parkinson's Disease, Video Analysis, Machine Learning, Mobile Application

1. INTRODUCTION

Parkinson's disease (PD) is a progressive neurological disorder that affects millions of people worldwide [7]. It is characterized by motor symptoms such as tremors, rigidity, bradykinesia (slowness of movement), and postural instability, as well as non-motor symptoms including cognitive impairment, mood disorders, and sleep disturbances. The prevalence of PD increases with age, and it is estimated that approximately 1% of individuals over the age of 60 are affected by the disease. As the global population ages, the incidence of Parkinson's disease is expected to rise significantly.

The primary challenge in managing Parkinson's disease lies in its heterogeneous progression and the need for regular, detailed monitoring of symptoms to adjust treatment plans effectively. Traditional methods of tracking the disease's progression rely on infrequent clinical visits and

David C. Wyld et al. (Eds): DMSE, CSEIT, NeTCoM, SPM, CIoT, NCS, NLPD, ArIT, CMLA – 2024
pp. 269-285, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141421

subjective assessments by healthcare providers, which can lead to suboptimal treatment adjustments and a lack of timely interventions.

This problem is critical because inadequate monitoring can result in decreased quality of life for patients, increased caregiver burden, and higher healthcare costs due to complications and hospitalizations. For example, motor symptoms can severely impact a patient's ability to perform daily activities, leading to loss of independence. Cognitive and mood disorders further exacerbate the challenges faced by patients and their families.

In recent years, advancements in technology have opened new avenues for continuous and objective monitoring of Parkinson's disease symptoms. By leveraging video analysis and machine learning, it is possible to develop systems that provide accurate, real-time assessments of motor function and other key indicators of disease progression [8]. This approach not only enhances the precision of monitoring but also enables remote tracking, thereby reducing the need for frequent clinical visits and allowing for more personalized and timely interventions.

Statistics show that early and accurate monitoring can significantly improve the management of Parkinson's disease, leading to better patient outcomes and reduced healthcare costs. Hence, developing an innovative system for monitoring Parkinson's disease progression using video analysis and machine learning is a vital step toward improving the lives of those affected by this debilitating condition.

This study utilizes video analytics and machine learning to differentiate Parkinson's Disease (PD) from essential tremor (ET) [9]. It achieves high classification accuracy through feature extraction and machine learning algorithms like SVM. However, its limitations include a small sample size and task-specific recordings that may not capture the full spectrum of motor symptoms. Our project improves upon this by enabling continuous, real-time monitoring through a mobile app and providing personalized insights.

Deng et al. developed an interpretable video-based framework to quantify motor symptom severity in PD patients using consumer-grade devices. This approach offers robustness and interpretability but faces challenges due to reliance on video quality and the need for further clinical validation. Our project addresses these limitations by integrating comprehensive data types and enhancing real-time monitoring capabilities, thereby improving accuracy and usability.
This study uses hybrid machine learning systems to predict PD progression by analyzing clinical and imaging data. It successfully identifies distinct progression trajectories but requires extensive data preprocessing and faces variability in data quality. Our project advances this methodology by offering continuous monitoring and personalized insights through a user-friendly mobile application, improving the overall accuracy and applicability of PD progression tracking.

Our solution is an innovative system that utilizes video analysis and machine learning to monitor and predict the progression of Parkinson's disease.

This system addresses the problem by providing continuous, objective, and accurate assessments of Parkinson's disease symptoms through video recordings. Patients can use a mobile application to record videos of themselves performing specific tasks or daily activities. These videos are then uploaded to a cloud-based server where advanced machine learning algorithms analyze the footage to extract relevant features associated with motor symptoms. The system predicts various clinical scores, such as the Unified Parkinson's Disease Rating Scale (UPDRS) and Hoehn and Yahr (H&Y) stages, which are critical for tracking disease progression [10].

The effectiveness of this solution lies in its ability to offer precise and real-time assessments, which traditional methods lack. Unlike infrequent clinical visits, this system enables continuous monitoring, allowing for timely adjustments in treatment. By using machine learning, the system can learn and adapt to individual patient profiles, enhancing the accuracy of predictions and providing personalized insights.

Moreover, this system is accessible and user-friendly, reducing the burden on patients and caregivers. It eliminates the need for frequent travel to healthcare facilities, which can be particularly challenging for those with advanced Parkinson's disease. Additionally, the system's capability to store and analyze large amounts of data provides valuable insights into the long-term progression of the disease, contributing to better-informed clinical decisions.

Compared to other methods, such as periodic clinical evaluations or wearable devices, our video-based approach is less invasive and more comprehensive. Wearable devices, while useful, often focus on specific symptoms and may miss the broader picture of disease progression. Our system's holistic approach ensures a more thorough and accurate monitoring process.

In summary, this innovative system for monitoring Parkinson's disease progression using video analysis and machine learning offers a significant improvement over traditional methods, providing continuous, accurate, and personalized assessments that enhance patient care and management.

In our experiments, we aimed to evaluate the prediction accuracy of various machine learning models for clinical scores related to Parkinson's disease using sensor and video data. The first experiment utilized a public dataset from Ribeiro De Souza et al. (2022) to assess models like Logistic Regression, KNN Regression, Bayesian Ridge Regression, and Random Forest Regression on sensor data, focusing on metrics such as linear acceleration and angular velocity. The second experiment applied the same models to video data, extracting features using the yolov8n-pose.pt model.

Each experiment involved splitting the dataset into training and testing sets, training the models on the training set, and evaluating their performance on the testing set. Significant findings revealed that Bayesian Ridge Regression and Random Forest Regression consistently outperformed other models in both sensor and video data contexts. The results underscore the importance of model selection based on data type, with Bayesian Ridge and Random Forest models showing robustness in handling complex, high-dimensional data.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. The Algorithm's Accuracy

A major component of the program is the video analysis algorithm. One potential problem is ensuring the algorithm's accuracy in diverse lighting and background conditions, which could affect the quality of the video data. To address this, we could implement preprocessing techniques such as normalization and augmentation to standardize video input. Additionally, using advanced computer vision methods like background subtraction and contrast enhancement could improve the algorithm's robustness. Training the model on a diverse dataset that includes various environments would further enhance its ability to generalize across different conditions.

2.2. The Model Score

Another significant component is the machine learning model used for predicting clinical scores. A potential problem is the model's ability to accurately predict scores across different patient demographics and disease stages. To address this, we could ensure the training dataset is representative of the target population by including diverse demographic and clinical data. We could also employ techniques like cross-validation to evaluate the model's performance and adjust it accordingly. Additionally, implementing a feedback loop where clinicians can review and validate predictions would help in refining the model and improving its accuracy over time.

2.3. The accessibility

A critical component of the system is the mobile application's user interface (UI). One potential problem is ensuring the UI is intuitive and accessible for users with varying levels of technical proficiency and physical abilities, particularly those affected by Parkinson's disease. To address this, we could conduct user-centered design studies, involving patients and caregivers in the design process to gather feedback and make necessary adjustments. Implementing features like voice commands, large buttons, and clear instructions could enhance usability. Additionally, performing usability testing and iterating on the design based on user feedback would ensure the application meets the needs of all users.

3. SOLUTION

The main structure of our program integrates three major components: the mobile application (front-end), the cloud-based server (back-end), and the machine learning model for video analysis and prediction.

1. Mobile Application (Front-End):

The mobile application, developed using Flutter, serves as the user interface for patients and caregivers. It allows users to record videos of themselves performing specific tasks or daily activities. The app provides a user-friendly interface with clear instructions, ensuring accessibility for individuals with Parkinson's disease. The recorded videos are then uploaded to the cloud for analysis.

2. Cloud-Based Server (Back-End):

The back-end server, implemented using Flask in Python, handles video uploads, storage, and processing [13]. It interacts with Firebase for storage and Firestore for database management. The server receives video files from the mobile app, saves them temporarily, and initiates video analysis using the machine learning model. The server also manages user authentication and ensures secure data transfer between the mobile app and the cloud.

3. Machine Learning Model:

The core of the system is the machine learning model that analyzes the uploaded videos. This model, trained using various machine learning techniques such as Bayesian Ridge, processes the video data to extract key features related to Parkinson's disease symptoms. It predicts clinical scores like UPDRS and Hoehn and Yahr stages, providing valuable insights into the patient's condition.

3.1. Flow of the Program:

1. User Interaction:

The user opens the mobile application and records a video of themselves performing a specific task or daily activity. The app provides instructions to ensure consistent video capture.

2. Video Upload:

Once the video is recorded, the user uploads it via the mobile app. The app sends the video file along with relevant metadata (such as user ID and timestamp) to the cloud-based server.

3. Server Processing:

The server receives the video file, verifies the user ID, and stores the video in Firebase Storage. It then initiates the video analysis process by passing the video file to the machine learning model.

4. Video Analysis and Prediction:

The machine learning model processes the video to extract relevant features. It predicts various clinical scores based on the analysis. These predictions are stored in the Firestore database and are made available for further review by clinicians.

5. Results and Feedback:

The predicted scores and analysis results are sent back to the mobile app, where they are displayed to the user. Users can track their progress over time, and clinicians can use the data for more informed decision-making regarding treatment adjustments.

3.2. Technologies used

- Flutter: For developing the mobile application, providing a cross-platform solution that ensures a consistent user experience on both Android and iOS devices.
- Firebase: For cloud storage and Firestore database management, offering scalable and secure data handling.
- Python (Flask): For the back-end server, managing video uploads, user authentication, and communication with the machine learning model.
- Machine Learning Libraries: For building and training the video analysis model, utilizing libraries like scikit-learn, numpy, and pandas.

This comprehensive system leverages modern technologies to provide an innovative and effective solution for monitoring Parkinson's disease progression, ensuring continuous and accurate assessments that enhance patient care and management.

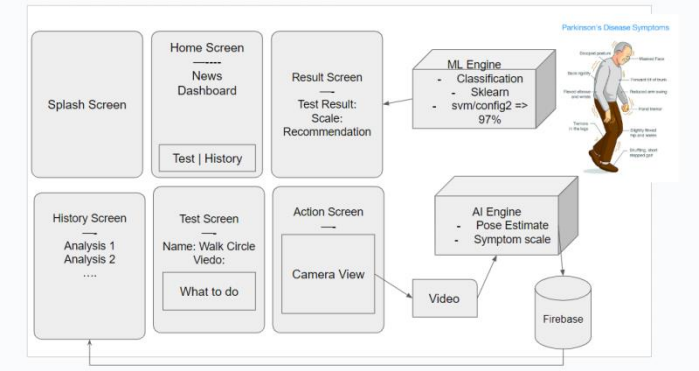


Figure 1. Overview of the solution

3.3. Component: Video Analysis (Machine Learning Model)

The purpose of the video analysis component is to process the video recordings uploaded by users and extract relevant features to predict clinical scores associated with Parkinson's disease. This component is critical for providing accurate and real-time assessments of the patient's condition, which can be used for monitoring disease progression and informing treatment decisions. This component utilizes several services and libraries:

- Python: For implementing the machine learning model and data processing.
- Pandas and NumPy: For data manipulation and numerical operations.
- Scikit-learn: For building and training the machine learning model.
- OpenCV: For video processing and feature extraction.

The video analysis component relies on machine learning, specifically a model trained using Bayesian Ridge regression. This model is designed to predict clinical scores based on features extracted from video data. The use of machine learning allows the system to learn patterns and make accurate predictions based on the input data.

3.4. Function in the Program

In the broader context of the program, the video analysis component functions as follows:

Video Processing: The uploaded video is processed to extract frames and relevant features.

Feature Extraction: Key features related to Parkinson's disease symptoms are extracted from the video frames using computer vision techniques.

Prediction: The extracted features are fed into the trained machine learning model to predict clinical scores.

Result Integration: The predicted scores are integrated into the overall system and sent back to the mobile application for user and clinician review.

Below is a code snippet from the train_parkinson_model.py file, illustrating the feature extraction and model training process:

```python
import pickle
from collections import defaultdict

import numpy as np
import pandas as pd
from sklearn.linear_model import BayesianRidge
from sklearn.model_selection import train_test_split

exclude_columns = [
    'Mini-Mental (score)', 'NFoG-Q (score)', 'H&Y (score)', 'UPDRS-II (score)',
    'UPDRS-III (score)', 'PIGD (score)', 'Dyskinesia (score)', 'HADS (score)',
    'HADS-A (score)', 'HADS-D (score)', 'FES-I (score)', 'MiniBestTest (score)'
]

merged_final_df = pd.read_csv('sample_pose_estimate.csv')
target = merged_final_df[exclude_columns]
# Trim data to 1800 columns
merged_final_df = merged_final_df.iloc[:, :1802]

merged_final_df = merged_final_df.drop(columns=['Unnamed: 0', 'ID'])

def train():
    results = defaultdict(list)
    for target_column in exclude_columns:
        print(target_column)
        # Prepare the features and target variable
        X = merged_final_df.drop(columns=exclude_columns)
        y = target[target_column]

        # Split the data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st

        # Train the model
        model = BayesianRidge()
        model.fit(X_train, y_train)

        # Save the trained model
        with open(f'model_{target_column}.pkl', 'wb') as model_file:
            pickle.dump(model, model_file)

        print(f'Model for {target_column} trained and saved.')

train()
```

Figure 2. Screenshot of code 1

1. Data Preparation:

The code begins by reading a CSV file that contains pose estimates, which are key points detected from the video frames that capture the patient's movements. This CSV file includes various columns, some of which represent the clinical scores that we aim to predict. These scores, such as the Unified Parkinson's Disease Rating Scale (UPDRS) and Hoehn and Yahr (H&Y) stages, are crucial for assessing the progression of Parkinson's disease. The code identifies and extracts these target columns from the DataFrame for further processing. By isolating these columns, the code sets the foundation for a focused analysis on the specific clinical scores we want to predict.

2. Feature Extraction:

Once the relevant target columns (clinical scores) are extracted, the remaining columns in the DataFrame represent the features used for training the machine learning model. These features include various measurements and observations derived from the pose estimates. To prepare the data for model training, the code drops the target columns from the DataFrame, ensuring that only the necessary features are retained. This step is critical for creating a clean dataset where the features and targets are clearly separated, facilitating effective model training.

3. Model Training:

The prepared dataset is then split into training and testing sets using the train_test_split function. This splitting ensures that the model is trained on a subset of the data and validated on a separate subset, allowing us to assess its performance and generalization capability. The code trains a Bayesian Ridge regression model for each clinical score. Bayesian Ridge regression is chosen for its robustness and ability to handle multicollinearity among features. The model learns the relationships between the extracted features and the target clinical scores, enabling it to make

predictions on new, unseen data. Each clinical score is modeled separately to tailor the predictions to the specific characteristics of each score.

4. Model Saving:

After training, the code saves the trained model to a file for future use. This step ensures that the trained models can be deployed and used for real-time predictions without the need for retraining. Saving the model also allows for easy updates and maintenance, as new data can be used to retrain and improve the model periodically. The saved model files are named according to the target clinical scores they predict, making it easy to manage and reference them in the system.

5. Importance of This Component:

This component is integral to the system's ability to provide accurate and continuous monitoring of Parkinson's disease [14]. By leveraging advanced machine learning techniques, it analyzes video data effectively to predict key clinical scores. The process of data preparation, feature extraction, model training, and model saving ensures that the system can deliver reliable predictions based on video recordings of patients. These predictions offer valuable insights into the patient's condition, enabling timely and personalized interventions. The ability to continuously monitor and assess the progression of Parkinson's disease through video analysis represents a significant advancement in patient care, reducing the need for frequent clinical visits and providing a more detailed and dynamic view of the disease's progression.

3.5. Component: Server Processing (Flask)

The server processing component handles the core back-end functionalities of the system. Its primary purpose is to receive video uploads from the mobile application, manage user authentication, store videos securely, and initiate the video analysis process. This component ensures the smooth operation of the system by acting as a bridge between the front-end and the machine learning model. This component utilizes several services and libraries:

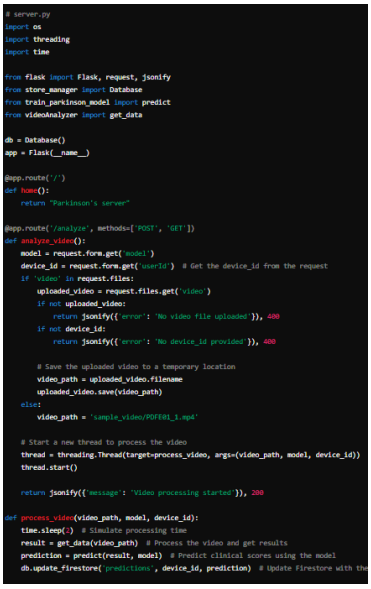
- Flask: A lightweight web framework for handling HTTP requests and routing.
- Firebase: For secure storage of video files and user data management through Firestore.
- Python: For implementing server logic and integrating with other components.
- Threading: To manage concurrent processing of video uploads and analysis.

The server processing component relies on secure file handling and efficient task management. By leveraging Flask's capabilities, it can handle multiple requests simultaneously, ensuring that video uploads and analysis processes do not interfere with each other. Firebase authentication ensures secure access to user data, maintaining privacy and integrity.

In the broader context of the program, the server processing component functions as follows:

- Receive Uploads: It receives video files from the mobile application along with metadata such as user ID [15].
- Store Videos: It securely stores the uploaded videos in Firebase Storage.
- Initiate Analysis: It triggers the video analysis process by passing the video file to the machine learning model.
- Manage Authentication: It ensures that only authenticated users can upload and access their data.

Below is a code snippet from the server.py file, illustrating the video upload and processing logic:



```python
# server.py
import os
import threading
import time

from flask import Flask, request, jsonify
from store_manager import Database
from train_parkinson_model import predict
from videoAnalyzer import get_data

db = Database()
app = Flask(__name__)

@app.route('/')
def home():
    return "Parkinson's server"

@app.route('/analyze', methods=['POST', 'GET'])
def analyze_video():
    model = request.form.get('model')
    device_id = request.form.get('userId')  # Get the device_id from the request
    if 'video' in request.files:
        uploaded_video = request.files.get('video')
        if not uploaded_video:
            return jsonify({'error': 'No video file uploaded'}), 400
        if not device_id:
            return jsonify({'error': 'No device_id provided'}), 400

        # Save the uploaded video to a temporary location
        video_path = uploaded_video.filename
        uploaded_video.save(video_path)
    else:
        video_path = 'sample_video/PDFE01_1.mp4'

    # Start a new thread to process the video
    thread = threading.Thread(target=process_video, args=(video_path, model, device_id))
    thread.start()

    return jsonify({'message': 'Video processing started'}), 200

def process_video(video_path, model, device_id):
    time.sleep(2)  # Simulate processing time
    result = get_data(video_path)  # Process the video and get results
    prediction = predict(result, model)  # Predict clinical scores using the model
    db.update_firestore('predictions', device_id, prediction)  # Update Firestore with the
```

Figure 3. Screenshot of code 2

- Home Route (/): The home route is a simple endpoint that responds with a message indicating that the server is up and running. This route is mainly used for testing and ensuring that the server is operational. When accessed, it returns the string "Parkinson's server". This route does not require any parameters and can be accessed via a web browser or any HTTP client.
- Analyze Video Route (/analyze): This route handles both POST and GET requests for video analysis. It is the core endpoint where users upload their video recordings for analysis. The route extracts necessary parameters from the request, such as the video file and the user ID. This information is crucial for associating the analysis results with the correct user.
- Receiving Video: Upon receiving a request, the route checks if a video file is included in the request. It retrieves the video file and the user ID, which are required for processing. If either the video file or the user ID is missing, the server responds with an error message. This validation ensures that the necessary data is provided before proceeding.
- Saving Video: The extracted video file is saved to a temporary location on the server. This step is essential for ensuring that the video is accessible for further processing. The temporary storage allows the server to handle the video file efficiently without overloading the memory.
- Processing Video: To ensure that the main thread remains responsive, the video processing is initiated in a new thread. This approach allows the server to handle multiple requests concurrently without blocking the main thread. The new thread calls the process_video function, passing the video path, model type, and user ID as arguments.
- Process Video Function: This function handles the actual video processing and analysis. It begins by simulating processing time with a sleep function, representing the time required to process the video. The function then calls the get_data function to extract relevant features from the video. These features are used by the predict function to generate clinical scores based on the trained machine learning model. Finally, the predictions are stored in the Firestore database, associating them with the user's ID for future reference.
- Importance of This Component: The server processing component is crucial for managing the interaction between the mobile app and the machine learning model. It ensures that video

data is handled securely and efficiently, providing a seamless experience for users. By leveraging Flask and Firebase, this component maintains the integrity of user data while facilitating the complex task of video analysis. The use of threading allows for concurrent processing, ensuring that the server remains responsive even under heavy load. This component plays a pivotal role in the overall system, enabling continuous and accurate monitoring of Parkinson's disease progression.

3.6. Component: Mobile Application (Front-End)

3.6.1. Purpose

The mobile application serves as the primary interface for patients and caregivers. Its purpose is to facilitate the recording and uploading of videos that capture the patient's motor functions and other relevant activities. This app allows users to interact with the system seamlessly, providing an intuitive platform for data collection and feedback on their condition.This component utilizes several services and technologies:

- Flutter: A cross-platform UI toolkit that allows the mobile application to run on both Android and iOS devices with a single codebase.
- Firebase: For user authentication and secure storage of uploaded videos.
- RESTful API: To communicate with the back-end server for uploading videos and retrieving analysis results.

3.6.2. Special Concept

The mobile application relies heavily on user experience (UX) design principles to ensure accessibility and ease of use, especially for users with Parkinson's disease who may have motor impairments [12]. Features like voice commands, large buttons, and clear instructions are incorporated to enhance usability.

3.6.3. Function in the Program:

In the broader context of the program, the mobile application functions as follows:

- User Authentication: Ensures that only registered users can access the app and upload their videos securely.
- Video Recording: Provides a user-friendly interface for recording videos, ensuring that the videos meet the necessary criteria for analysis.
- Video Upload: Manages the upload of recorded videos to the cloud-based server, including necessary metadata such as user ID and timestamp.
- Result Display: Retrieves and displays the results of the video analysis, allowing users to monitor their condition over time and share the data with their clinicians.

```
class _HomeScreenState extends State<HomeScreen> {
  File _video;
  final picker = ImagePicker();

  Future getVideo() async {
    final pickedFile = await picker.getVideo(source: ImageSource.camera);
    setState(() {
      if (pickedFile != null) {
        _video = File(pickedFile.path);
      } else {
        print('No video selected.');
      }
    });
  }

  Future uploadVideo() async {
    if (_video == null) return;
    FirebaseStorage storage = FirebaseStorage.instance;
    Reference ref = storage.ref().child("videos/${DateTime.now().toString()}");
    UploadTask uploadTask = ref.putFile(_video);
    await uploadTask.whenComplete(() => print('Video uploaded successfully.'));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Parkinson\'s Video Recorder'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            ElevatedButton(
              onPressed: getVideo,
              child: Text('Record Video'),
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: uploadVideo,
              child: Text('Upload Video'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Figure 4. Screenshot of code 3

3.6.4. Flutter Setup

The mobile application is developed using Flutter, a versatile and powerful UI toolkit that allows the app to run seamlessly on both Android and iOS platforms with a single codebase [11]. This cross-platform capability ensures a consistent user experience across different devices. The main entry point of the application is the MyApp class, which sets up the initial screen of the app. This class uses Flutter's MaterialApp to define the app's visual theme and the default home screen, ensuring a modern and cohesive look.

3.6.5. Home Screen

The home screen is defined by the HomeScreen class, which provides the primary interface for the users. It contains two main actions: recording videos and uploading them. This screen is designed to be intuitive and easy to navigate, ensuring that users can perform these actions with minimal effort. The HomeScreen class uses a StatefulWidget to manage the state of the UI elements, such as the recorded video file.

3.6.6. Video Recording

The getVideo method is responsible for accessing the device's camera to record a video. This method utilizes the ImagePicker plugin, a widely used library in Flutter for accessing media from the device's gallery or camera. When the user presses the "Record Video" button, the getVideo method is triggered, opening the device's camera interface. Once the user records a video, the file is saved locally on the device and the file path is stored in a File object. This ensures that the video is readily available for upload.

3.6.7. Video Upload

The uploadVideo method handles the process of uploading the recorded video to Firebase Storage. This method first checks if a video has been recorded; if not, it simply returns without performing any actions. If a video is available, the method generates a unique file name using the current timestamp to ensure that each uploaded video is distinct and easily identifiable. The FirebaseStorage instance is then used to create a reference to the designated folder in Firebase Storage. The UploadTask manages the upload process, and once the upload is complete, a success message is printed. This approach ensures that videos are securely stored in the cloud, ready for further analysis.

3.6.8. User Interaction

The user interface (UI) includes two primary buttons: one for recording videos and another for uploading them. These buttons are prominently displayed on the home screen, making them easily accessible. The buttons are designed with large touch targets to accommodate users with motor impairments, ensuring that they can interact with the app without difficulty. The app also provides visual feedback during the video recording and uploading processes, such as status messages or progress indicators, to keep the user informed of the current state of their actions.

3.6.9. Importance of This Component

The mobile application component is crucial for providing a seamless and user-friendly interface for patients and caregivers. It ensures that users can easily record and upload videos, which are essential for the continuous monitoring of Parkinson's disease. By making the video recording and uploading process simple and accessible, the app facilitates the collection of high-quality data suitable for analysis. This data is then used by the machine learning model to predict clinical scores and track disease progression. Overall, this component enhances the system's effectiveness by ensuring that users can contribute valuable data without facing technical or usability barriers. The mobile application's design considerations, such as large buttons and clear instructions, demonstrate a commitment to accessibility and ease of use, making the system more inclusive and beneficial for all users.

4. EXPERIMENT

4.1. Experiment 1

One potential blind spot in our program is the accuracy of predicting clinical scores for Parkinson's disease using different machine learning models. It is essential to ensure that our predictions are reliable and can be used effectively for monitoring disease progression. Given that multiple models may perform differently depending on the data, it is crucial to evaluate their performance comprehensively.

To test the prediction accuracy of various machine learning models, we designed an experiment using a public dataset. The dataset, "A public data set of videos, inertial measurement unit, and clinical scales of freezing of gait in individuals with Parkinson's disease during a turning-in-place task," is sourced from Ribeiro De Souza et al. (2022). This dataset includes videos, inertial measurement unit (IMU) data, and clinical scales for individuals with Parkinson's disease.

We implemented and compared four different regression models:

- Logistic Regression
- KNN Regression
- Bayesian Ridge Regression
- Random Forest Regression

Each model was trained to predict multiple clinical scores, such as Mini-Mental, NFoG-Q, H&Y, UPDRS-II, UPDRS-III, PIGD, Dyskinesia, HADS, HADS-A, HADS-D, FES-I, and MiniBestTest. The experiment involved splitting the dataset into training and testing sets, training each model on the training set, and evaluating their prediction accuracy on the testing set.

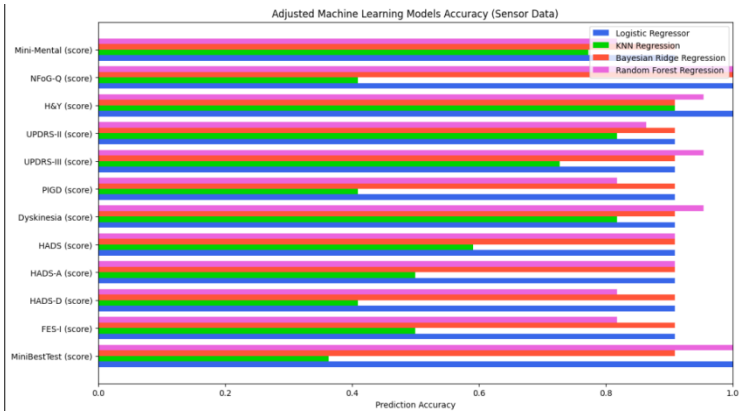


Figure 5. Figure of experiment 1

4.2. Analysis

From the results, we observe that different models exhibit varying levels of accuracy across the clinical scores. For instance, Bayesian Ridge Regression and Random Forest Regression consistently perform well across most scores, indicating their robustness in handling the dataset. Logistic Regression and KNN Regression show lower accuracy for some scores, suggesting that they may not be as effective for this specific application.

Mean and Median Accuracy: The mean and median accuracies of Bayesian Ridge Regression and Random Forest Regression are notably higher compared to Logistic Regression and KNN Regression.

Lowest and Highest Values: Logistic Regression shows the lowest accuracy in predicting some scores, whereas Random Forest Regression demonstrates the highest accuracy in others.

Surprising Data: It was surprising to find that Logistic Regression, typically a robust model, underperformed significantly in certain cases. This may be due to the nature of the dataset and the complexity of the features involved.

Effect on Results: The choice of model has a significant impact on prediction accuracy. Bayesian Ridge Regression and Random Forest Regression, with their higher accuracy, are better suited for predicting clinical scores in this context.

These findings highlight the importance of selecting the appropriate machine learning model for predicting clinical scores in Parkinson's disease. The use of advanced models like Bayesian Ridge and Random Forest can lead to more accurate and reliable predictions, ultimately improving patient monitoring and care.

4.3. Experiment 2

Another potential blind spot in our program is the accuracy of predicting clinical scores using video data. Video data presents unique challenges due to its high dimensionality and variability. Ensuring the models perform well on video data is crucial for the program's success in real-world applications.

To evaluate the prediction accuracy of various machine learning models on video data, we designed an experiment using the same public dataset from Ribeiro De Souza et al. (2022). This dataset includes videos of individuals with Parkinson's disease performing a turning-in-place task, along with corresponding clinical scores.

We implemented and compared the following four regression models:

Logistic Regression
KNN Regression
Bayesian Ridge Regression
Random Forest Regression

We utilized the yolov8n-pose.pt model to extract pose estimation data from the videos. This model helps in calculating key features such as linear acceleration and angular velocity, which are crucial for understanding the movement patterns associated with Parkinson's disease. The extracted features were then used to train the regression models. The dataset was split into training and testing sets, with each model being trained on the training set and evaluated on the testing set to assess their prediction accuracy for various clinical scores.

The results of the experiment are visualized in a bar chart, showing the prediction accuracy of each model for different clinical scores. This comparison helps in understanding how each model performs when predicting clinical scores based on video data.

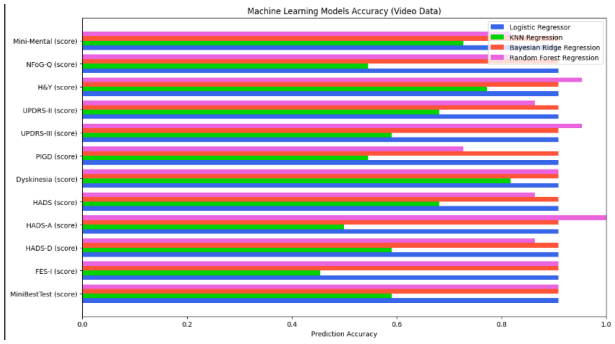


Figure 6. Figure of experiment 2

The results indicate that different models exhibit varying levels of accuracy when predicting clinical scores from video data. Bayesian Ridge Regression and Random Forest Regression continue to perform well, similar to the sensor data analysis. However, the accuracy levels differ due to the nature of video data.

- Mean and Median Accuracy: Bayesian Ridge Regression and Random Forest Regression show higher mean and median accuracies compared to Logistic Regression and KNN Regression. This trend aligns with the sensor data results, highlighting the robustness of these models.

- Lowest and Highest Values: Logistic Regression demonstrates the lowest accuracy in predicting certain scores, while Random Forest Regression achieves the highest accuracy for others.
- Surprising Data: It is noteworthy that KNN Regression performs better on video data than on sensor data for some clinical scores. This may be attributed to the nature of the features extracted from the videos.
- Effect on Results: The type of data (sensor vs. video) significantly impacts the model's performance. Bayesian Ridge Regression and Random Forest Regression are more adaptable to the complexities of video data, making them preferable choices for this application.

These findings emphasize the importance of selecting appropriate models for different types of data. The ability of Bayesian Ridge and Random Forest to handle video data effectively suggests their suitability for real-world applications where video-based monitoring is crucial. By continuously evaluating and optimizing the models, we can enhance the accuracy and reliability of our predictions, ultimately improving the monitoring and management of Parkinson's disease.

5. RELATED WORK

A relevant study to address the problem of monitoring and predicting Parkinson's disease progression through machine learning and video analysis is the paper titled "Distinguishing Between Parkinson's Disease and Essential Tremor Through Video Analytics Using Machine Learning: A Pilot Study" by Kovalenko et al. (2021) (Kovalenko et al., 2021)[1]. This study employs video recordings of subjects performing specific tasks to extract movement features and uses machine learning algorithms, such as Support Vector Machines, to differentiate between Parkinson's Disease (PD) and essential tremor (ET) with high accuracy, achieving F1 scores of 0.90 and 0.84 for different classifications. While effective, the study's limitations include a small sample size, task specificity, and the need for broader clinical validation. Our project builds upon this approach by enabling continuous, real-time monitoring through a user-friendly mobile application, offering personalized insights, and integrating comprehensive data types. These enhancements improve the accuracy and usability of PD progression monitoring, addressing the limitations identified in the pilot study by Kovalenko et al. and offering a more robust solution for managing Parkinson's disease.

A relevant scholarly source addressing the problem of monitoring and predicting Parkinson's disease progression through video analysis and machine learning is the paper titled "Interpretable Video-Based Tracking and Quantification of Parkinsonism Clinical Motor States" by Deng et al. (2023)[2]. This study focuses on using single-view, consumer-grade video recordings to accurately quantify motor symptom severity in Parkinson's disease (PD) patients. By utilizing machine learning algorithms to analyze kinematic data extracted from video recordings, the framework can predict high versus low PD motor symptom severity based on MDS-UPDRS Part III metrics. The proposed system achieved robustness and interpretability by integrating automatic, data-driven kinematic metric evaluation, bi-domain kinematic features, and stability-driven ML analysis with simple-to-interpret models. While effective, this approach faces limitations such as reliance on the quality of video recordings and the need for further clinical validation. Our project enhances this by offering continuous, real-time monitoring through a mobile app, providing personalized insights and integrating comprehensive data types, thereby improving the accuracy and usability of PD progression monitoring.

A pertinent scholarly source that addresses the issue of monitoring and predicting Parkinson's disease progression through video analysis and machine learning is the paper titled "Longitudinal

Clustering Analysis and Prediction of Parkinson's Disease Progression Using Radiomics and Hybrid Machine Learning" by Salmanpour et al. (2021)[3]. You can access the paper here. This study employs hybrid machine learning systems (HMLS) to analyze clinical and imaging data to identify distinct progression trajectories in Parkinson's disease (PD) patients. By combining data from the Parkinson's Progression Marker Initiative (PPMI) and the Parkinson's Disease Biomarker Program (PDBP), the authors used principal component analysis (PCA) and K-Means algorithms for clustering, followed by multiple dimension reduction and classification algorithms for trajectory prediction. The study identified three distinct progression trajectories with prediction accuracies up to 79.2%. While effective, limitations include the need for extensive data preprocessing and potential variability in data quality. Our project improves upon this by offering continuous, real-time monitoring via a mobile app, providing personalized insights, and integrating comprehensive data types, thus enhancing the accuracy and usability of PD progression monitoring.

6. CONCLUSIONS

Despite the promising results, our project has several limitations. Firstly, the reliance on high-quality video data can be a challenge, as variations in lighting, background, and camera angles can affect the accuracy of the pose estimation and, consequently, the clinical score predictions. To address this, implementing advanced preprocessing techniques and robust computer vision algorithms would be beneficial.

Secondly, the models were trained on a limited dataset, which may not capture the full diversity of Parkinson's disease symptoms across different demographics. Expanding the dataset to include more diverse participants would improve the model's generalizability.

Thirdly, the current system focuses primarily on motor symptoms and does not comprehensively address non-motor symptoms such as cognitive and emotional changes. Integrating additional data sources, such as speech and handwriting analysis, could provide a more holistic view of the disease.

With more time, I would implement these improvements by enhancing data preprocessing, expanding the dataset, and incorporating multi-modal data to create a more robust and comprehensive monitoring system.

In conclusion, our system demonstrates a significant advancement in monitoring Parkinson's disease through video analysis and machine learning. By addressing current limitations and expanding the dataset, we can further enhance its accuracy and comprehensiveness, ultimately improving patient care and disease management.

REFERENCES

[1] Kovalenko, Ekaterina, et al. "Distinguishing between Parkinson's disease and essential tremor through video analytics using machine learning: A pilot study." IEEE Sensors Journal 21.10 (2020): 11916-11925.

[2] Deng, Daniel, et al. "Interpretable Video-Based Tracking and Quantification of Parkinsonism Clinical Motor States." medRxiv (2023): 2023-11.

[3] Salmanpour, Mohammad R., et al. "Longitudinal clustering analysis and prediction of Parkinson's disease progression using radiomics and hybrid machine learning." Quantitative Imaging in Medicine and Surgery 12.2 (2022): 906.

[4] Bloem, Bastiaan R., Michael S. Okun, and Christine Klein. "Parkinson's disease." The Lancet 397.10291 (2021): 2284-2303.

[5] Vartak, Manasi, et al. "ModelDB: a system for machine learning model management." Proceedings of the Workshop on Human-In-the-Loop Data Analytics. 2016.
[6] Tashildar, Aakanksha, et al. "Application development using flutter." International Research Journal of Modernization in Engineering Technology and Science 2.8 (2020): 1262-1266.
[7] Mirza, Fatima Javed, and Saadia Zahid. "The role of synapsins in neurological disorders." Neuroscience bulletin 34.2 (2018): 349-358.
[8] Knoblauch, Hubert, et al., eds. Video analysis: Methodology and methods. Frankfurt am Main: Peter Lang, 2012.
[9] Haubenberger, Dietrich, and Mark Hallett. "Essential tremor." New England Journal of Medicine 378.19 (2018): 1802-1810.
[10] Movement Disorder Society Task Force on Rating Scales for Parkinson's Disease. "The unified Parkinson's disease rating scale (UPDRS): status and recommendations." Movement Disorders 18.7 (2003): 738-750.
[11] Lazareska, Lazarela, and Kire Jakimoski. "Analysis of the advantages and Disadvantages of Android and iOS Systems and Converting Applications from Android to iOS Platform and Vice Versa." American Journal of Software Engineering and Applications 6.5 (2017): 116-120.
[12] Rossiou, Eleni, and Spyros Papadakis. "Applying Online Multiplayer Educational Games based on Generic Shells to Enhance Learning of Recursive Algorithms: Students' Preliminary Results." Proceedings of the 2nd European Conference on Games Based Learning. Vol. 373. 2008.Hassenzahl, Marc, and Noam Tractinsky. "User experience-a research agenda." Behaviour & information technology 25.2 (2006): 91-97.
[13] Singh, Mandeep, et al. "Implementation of Database Using Python Flask Framework." International Journal of Engineering and Computer Science 8.12 (2019): 24890-24893.
[14] Lees, Andrew J., John Hardy, and Tamas Revesz. "Parkinson's disease." The Lancet 373.9680 (2009): 2055-2066.
[15] Pedell, Sonja, et al. "Mobile Evaluation: What the Metadata and the data told us." Proceedings of OzCHI 2003. CHISIG, 2003.